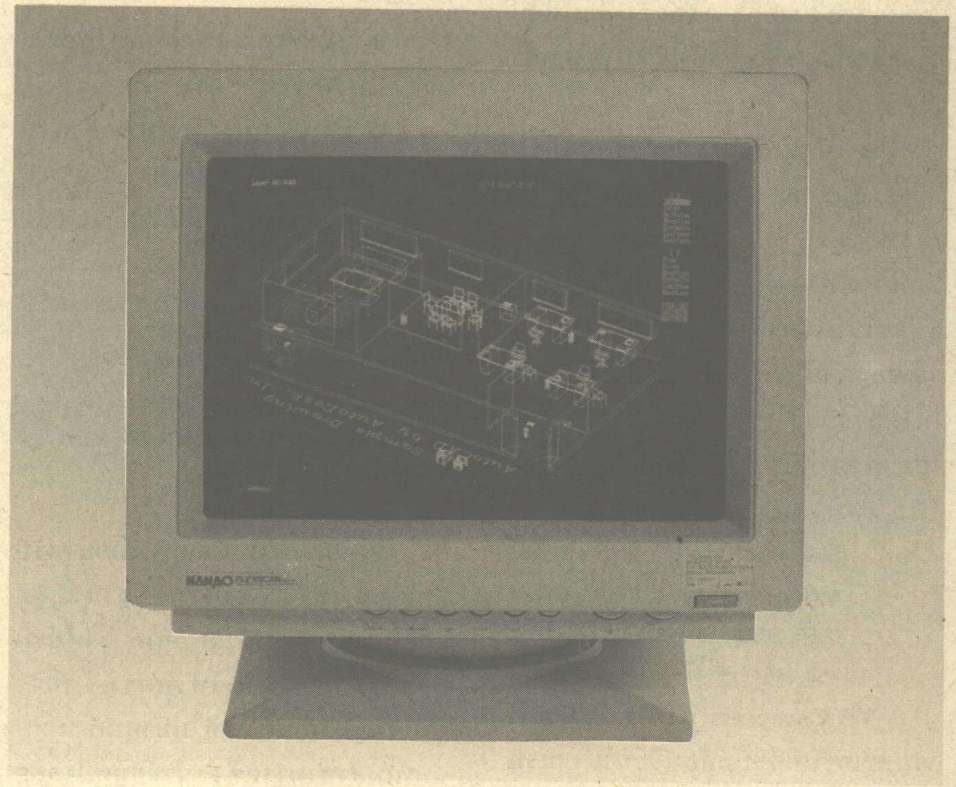
ชัชวาล ศุภเกษม

เทคนิคใหม่ในการเขียนข้อความบน AutoCAD

เสนอคำสั่งที่พัฒนาขึ้นด้วยโปรแกรม AutoLISP ซึ่งจะช่วยจัดการงานเขียนข้อความบน AutoCAD ให้สะดวกและลดเวลาในการทำงาน

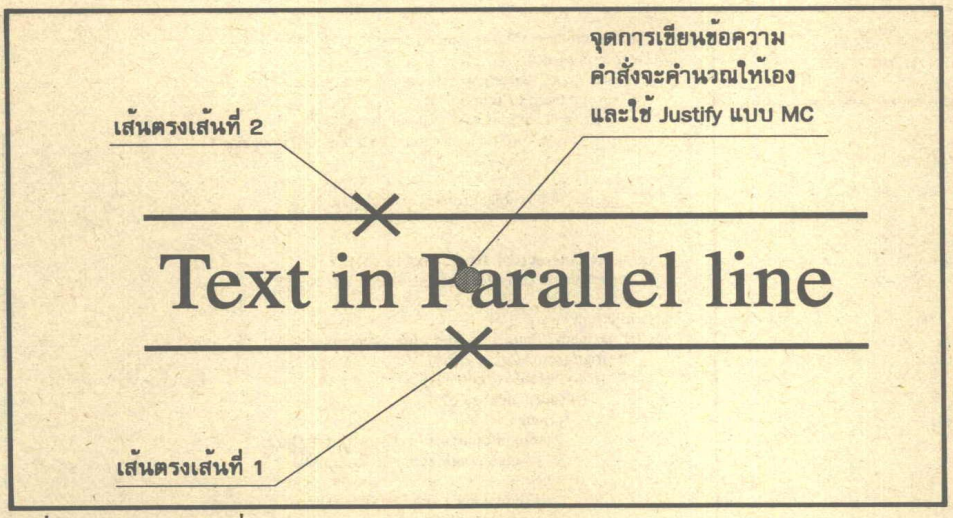


บทความนี้จะนำเสนอคำสั่งที่พัฒนาขึ้นด้วยโปรแกรม AutoLISP ซึ่งจะช่วยจัดการงานเขียนข้อความต่าง ๆ ใน AutoCAD ให้สะดวก และลดเวลาในการทำงาน โดยจะมีดังนี้ การเขียนข้อความระหว่างเส้นขนาน การเขียนตัวอักษรในเส้นตรงแบบอัตโนมัติ การเปลี่ยน CASE ของข้อความ และการแก้ไขข้อความ

การเขียนข้อความระหว่างเส้นขนาน

การเขียนแบบแปลนที่เกี่ยวกับการเขียนข้อความระหว่างเส้นขนาน เช่น งานที่มีส่วนแสดงแนวถนนที่ต้องมีการระบุชื่อเรียก หรือบริเวณอ้างอิง งานไฟฟ้าที่ต้องระบุชื่อระบบ หรือขนาดของช่องเดินสายไฟ (raceway เช่น Wireway, Ladder ฯ) งานเครื่องกลที่มีการระบุชนิดและขนาดของท่อแอร์ (air duct) งานสายพานลำเลียง (conveyor) ระบุรายละเอียด ของสายพานงานแบบแปลนต่าง ๆ ที่จะระบุชื่อแบบภายในเส้นคู่ขนาน ในการเขียนแบบด้วย AutoCAD จะเริ่มจากกำหนด Text style รันคำสั่ง Text หรือ Dtext ป้อนจุดตำแหน่งในการเขียน ความสูงอักษร กำหนดมุมหมุน ตามแนวเส้นขนาน แล้วเขียนข้อความที่ต้องการ สิ่งที่ต้องเสียเวลาในการกำหนดเห็นจะเป็นส่วนการกำหนดจุดการเขียน ให้อยู่กึ่งกลางพอดีของแนวเส้นขนานนั้น ๆ ในบทความนี้ขอนำเสนอคำสั่งที่ช่วยจัดการงานลักษณะนี้ โดยพัฒนาขึ้นในรูปโปรแกรม AutoLISP ชื่อว่า TINPL (Text IN Parallel Line)

การทำงานของคำสั่ง TINPL ผู้ใช้เพียงแต่เลือกเส้นขนานที่จะเขียนข้อความแทรกอยู่ระหว่างกลาง ป้อนค่าความสูงอักษร และตามด้วยข้อความที่ต้องการ ส่วนจุดการเขียนโปรแกรมจะคำนวณให้ซึ่งจะเป็นจุดกึ่งกลางระหว่างเส้นขนาน ที่อยู่ถัดไปจากจุดที่เลือกครั้งแรก โดยเป็น Justify แบบ MC (Middle Center) นอกจากนั้นโปรแกรมยังคำนวณค่ามุมหมุน (Rotation) ของการเขียนให้เองด้วย



รูปที่ 1 แสดงการใช้คำสั่ง TINPL

ก่อนอื่นขอให้ผู้อ่านได้สร้างไฟล์ TINPL.lsp นี้ ด้วยโปรแกรม Text editor ใด ๆ ที่ท่านมีอยู่ ดังแสดงตัวโปรแกรมในรูปที่ 2 จากนั้นทำการโหลดไฟล์ที่บรรทัด Command: ใน Auto-CAD ดังนี้

```
Command: (load "C:/lisp/tinpl")
```

โดย c:/lisp เป็น Path ที่เก็บไฟล์นี้อยู่ ขอให้สังเกตว่าใช้เครื่องหมาย forward slash (/) เป็นเครื่องหมายบอก Path ต่างจากในดอส ที่ใช้เครื่องหมาย back slash (\) ขอให้ระบุชื่อ Path ที่เก็บให้ตรงด้วย ถ้าไม่มีการพิมพ์ไฟล์ผิดพลาด หรือตกหล่นเลย จะมีข้อความที่บรรทัด Command: เป็น C:TINPL ซึ่งจะพร้อมใช้งาน โดยการพิมพ์ TINPL โดยตรงที่บรรทัด Command: ได้ สำหรับในกรณีที่พบกับข้อความแสดงความผิดพลาดขอให้กลับไปแก้ไขไฟล์นี้ให้ถูกต้อง แล้วทำการโหลดใหม่อีกครั้ง

ใช้คำสั่ง TINPL

เราจะมาทดสอบการทำงานของคำสั่ง TINPL ดังนี้

1. เริ่มจากเปิด Drawing ใหม่ แล้วทำการเขียนเส้นตรงเส้นหนึ่ง จากนั้นทำการ Offset เส้นตรงนี้ไปห่างจากเส้นเดิม 1 หน่วย เพื่อสร้างเป็นเส้นขนาน

2. ในตอนนี้ถ้ายังไม่มีการโหลดไฟล์ TINPL ให้กระทำก่อนตามวิธีการที่ได้กล่าวไว้ในข้างต้น จากนั้นที่บรรทัด Command: พิมพ์ TINPL กด Enter* (*ต่อไปจะละคำว่า "กด Enter" ออก ซึ่งผู้อ่านจะต้องกระทำด้วยทุกครั้ง เมื่อป้อนค่าครบแล้ว)

3. จะมีพรอมต์

```
1st Line, Select object:
```

ให้เลือกเส้นตรงที่เป็นเส้นขนานที่จะเขียนข้อความแทรกไว้ โดยจุดการเลือกจะถูกใช้เป็นตำแหน่งการเขียนอักษรแบบ Justify MC ที่จุดกึ่งกลางที่ถัดเข้าไประหว่างเส้นขนาน ในที่นี้ให้เลือกเส้นตรงที่ได้เขียนไว้เส้นหนึ่ง

ถ้าเลือกไม่ถูกวัตถุเลย จะมีข้อความเตือนว่า Must be select a line entity. และวนกลับไปที่พรอมต์เดิมให้เลือกใหม่ และถ้าเลือกวัตถุที่ไม่ใช่เส้นตรง (Line) จะมีข้อความเตือนว่า The entity must be a line. Try again. และวนกลับไปที่พรอมต์เดิมให้เลือกใหม่อีกครั้ง เป็นเช่นนี้จนกว่าจะเลือกได้ถูกต้อง

4. จะมีพรอมต์

```
2nd Line, Select object:
```

ให้เลือกเส้นตรงอีกเส้นที่เป็นเส้นขนานที่จะเขียนข้อความแทรก ในที่นี้เลือกเส้นตรงอีกเส้นที่เหลือ

เช่นเดียวกับการเลือกเส้นตรงในข้อ 3 จะมีข้อความเตือนในลักษณะเดียวกัน นอกจากนั้นยังมีสิ่งที่น่าสนใจกล่าวคือ ถ้าเลือกเส้นตรงซ้ำกับเส้นเดิมที่เลือกไว้ในตอนแรก จะมีข้อความเตือนว่า The 2nd line must not duplicate to 1st line. Try again. ซึ่งหมายถึงว่าเส้นที่สองนี้ต้องไม่ซ้ำกับเส้นที่หนึ่ง และจะวนกลับไปที่พรอมต์เดิม ให้เลือกใหม่อีกครั้ง

และถ้าการเลือกได้เส้นตรงที่ไม่ขนานกับเส้นตรงแรก จะมีข้อความเตือนว่า The 2nd line must be Parallel to 1st line. และจะวนกลับไปที่พรอมต์เดิม ให้เลือกใหม่อีกครั้ง เป็นเช่นนี้จนกว่าจะเลือกได้ถูกต้อง

5. จะมีพรอมต์

```
Text Height: < >
```

ให้ป้อนค่าความสูงของตัวอักษร ในที่นี้ป้อน 0.3

ถ้าค่าที่ป้อนนี้เกินกว่าระยะห่างของเส้นขนาน จะมีข้อความเตือนว่า Text Height must not exceeds the distance between 2 lines. และวนกลับไปที่พรอมต์เดิม ให้ป้อนค่าใหม่อีกครั้ง เป็นเช่นนี้จนกว่าจะได้ค่าที่น้อยกว่าระยะห่างของเส้นขนานนี้

6. จะมีพรอมต์

```
Text: < >
```

```
;;; TINPL
;;; Program to draw a text in a parallel line.
;;; Developed by S.Chatchawal, 1994.
;;;=============================================================
;; Read dxf
(defun dxf ( code e ) (cdr (assoc code (entget e))))
;; Determination 2 points in angle range [0,pi/2]U(3*pi/2,2*pi).
(defun angset ( p1 p2 / ang )
  (setq ang (angle p1 p2))
  (if (or (and (>= ang 0)(<= ang (/ pi 2.0)))
          (and (> ang (/ (* 3.0 pi) 2.0)) (< ang (* 2.0 pi)))
      )
      (list p1 p2)(list p2 p1)
  )
)
;; Vector from 3Dpoint p1 to p2.
(defun vector (p1 p2) (mapcar '- p2 p1))
;; Unit vector from 3Dpoint p1 to p2.
(defun 1vector (p1 p2 / d )
  (setq d (distance p1 p2))(mapcar '/ (vector p1 p2) (list d d d))
)
;; Dot product of 2 vectors
(defun dot ( u v ) (apply '+ (mapcar '* u v)))
;; Projection of 3Dpoint p on line entity e.
(defun ponl ( p e / p1 p2 p1p l1 d d1l)
  (setq p1 (dxf 10 e) p2 (dxf 11 e)
        p1p (vector p1 p) l1 (1vector p1 p2)
        d (dot l1 p1p) d1l (mapcar '* l1 (list d d d))
  )
  (mapcar '+ p1 d1l)
)
;; GDDEF (GetDist with DEFault value check subroutine)
;; Argument: a prompt string & a global variabe name.
;; Must be declare a global variable,gvar. Before use GDDEF.
(defun gddef ( prmpt gvar / bye var)
  (while (not bye)
    (princ (strcat "\n  " prmpt ":<"))(princ gvar)(princ "> ")
    (setq var (getdist))
    (if (= var nil)
        (if (/= gvar "")(setq bye T)
            (princ "\n!!! Now not a default value. Must be input value, first.")
        )
        (setq bye T gvar var)
    )
  )
  gvar
)
;; GSTDEF (GetString with DEFault value check subroutine)
;; Argument: a prompt string & a global variabe name.
;; Must be declare a global variable, gvar. Before use GSTDEF.
(defun gstdef ( prmpt gvar / bye var)
  (while (not bye)
    (princ (strcat "\n  " prmpt ":<"))(princ gvar)(princ "> ")
    (setq var (getstring T))
    (if (= var "")
        (if (/= gvar "")
            (setq bye T)
            (princ "\n!!! Now not a default value. Must be input value, first.")
        )
        (setq bye T gvar var)
    )
  )
  gvar
)
;; Main program
(setq #txth "" #txt "")
(defun c:tinpl (/ oldcmd oldlay oldtxt olderr stop
                  epl1 e1 p1 pl1 ang1 epl2 e2 p2 pl2 ang2 u1 v1 h d inpt
               )
  (setq oldcmd (getvar "cmdecho") oldlay (getvar "clayer")
        oldtxt (getvar "texteval") olderr *error*
  )
  (defun *error* (msg)
    (if (not (member msg '("console break" "Function cancelled")))
        (princ (strcat "Error: " msg))
    )
    (command "undo" "e" "u" "layer" "s" oldlay ""
             "texteval" oldtxt "cmdecho" oldcmd
    )
    )
    (setq *error* olderr)(princ)
  )
  (setvar "cmdecho" 0)(command "undo" "g")
  (while (not stop)
    (if (setq epl1 (entsel "\n 1st Line, Select object: "))
        (progn (setq e1 (car epl1))
            (if (= "LINE" (dxf 0 e1))(setq stop T)
                (princ "\n The entity must be a line. \n Try again.")
            )
        )
        (princ "\n No object found. \n Try again.")
    )
  )
  (setq pl1 (angset (dxf 10 e1) (dxf 11 e1))
        ang1 (angle (car pl1)(cadr pl1)) stop nil
  )
  (while (not stop)
    (if (setq epl2 (entsel "\n 2nd Line, Select object: "))
        (progn (setq e2 (car epl2))
            (if (= "LINE" (dxf 0 e2))
                (if (not (equal e1 e2))
                    (progn
                        (setq pl2 (angset (dxf 10 e2)(dxf 11 e2))
                              ang2 (angle (car pl2)(cadr pl2))
                        )
                        (if (equal ang1 ang2 0.00000001)
                            (setq stop T)
                            (princ "\n The 2nd line must be Parallel to 1st line. \n Try again.")
                        )
                    )
                    (princ "\The 2nd line  must not be duplicate to 1st line. \n Try again.")
                )
                (princ "\n The entity must be a line. \n Try again.")
            )
        )
        (princ "\n No object found. \n Try again.")
    )
  )
  (setq p1 (ponl (cadr epl1) e1) p2 (ponl (cadr epl2) e2)
        h (distance p2 (ponl p2 e1))  ; distance between L1 & L2.
        stop nil
  )
  (while (not stop)
    (setq #txth (gddef "Text Height" #txth))
    (if (< #txth h)(setq stop T)
        (princ
          (strcat
            "\n Text Height must not exceeds the distance between 2 lines."
            "\n Try again."
          )
        )

    )
  )
  (setq #txt (gstdef "Text" #txt)        ; Get message to draw.
        u1 (1vector (car pl1) (cadr pl1)) ; 1vector of L1.
        v1 (1vector p1 p2)     ; 1vector from selected point on L1 &L2.
        d (* (dot u1 v1)(distance p1 p2))       ; projection of V on L1.
        inpt (mapcar '/ (mapcar '+ p1 (mapcar '- p2 (mapcar '* (list d d d) u1)))
                        '(2.0 2.0 2.0)
             )
        ang1 (/ (* (/ (+ ang1 ang2) 2.0) 180.0) pi)
  )
  (command "layer" "s" (dxf 8 e1) "" "texteval" 1
           "text" "mc" inpt #txth ang1 #txt
           "layer" "s" oldlay "" "texteval" oldtxt
           "undo" "e" "cmdecho" oldcmd
  )
  (setq *error* olderr)(princ)
)
```

รูปที่ 2 ไฟล์โปรแกรม TINPL.lsp

ให้ป้อนข้อความที่ต้องการเขียน ในที่นี้ลองป้อน

```
Test TINPL
```

โปรแกรมจะทำการเขียนข้อความที่ป้อนนี้ แทรกอยู่ระหว่างกลางเส้นขนานที่เลือกไว้ โดยใช้ Justify แบบ MC ในจุดที่อยู่กึ่งกลางถัดเข้าไปในเส้นขนานจากจุดแรกที่เลือกเส้นตรงไว้

ข้อสังเกต

1. ข้อความที่เขียนจะอยู่บน Layer เดียวกับเส้นตรงที่เลือก

2. ไม่ว่าเส้นตรงจะอยู่ในแนวใด ๆ โปรแกรมจะกำหนดการเขียนอักษรเฉพาะมุมในช่วง [0, pi/2] กับ (−pi/2,0) (เป็นมุมในหน่วยเรเดียน) ทั้งนี้เพื่อให้อ่านแบบได้โดยไม่ต้องมีการหมุนแบบ

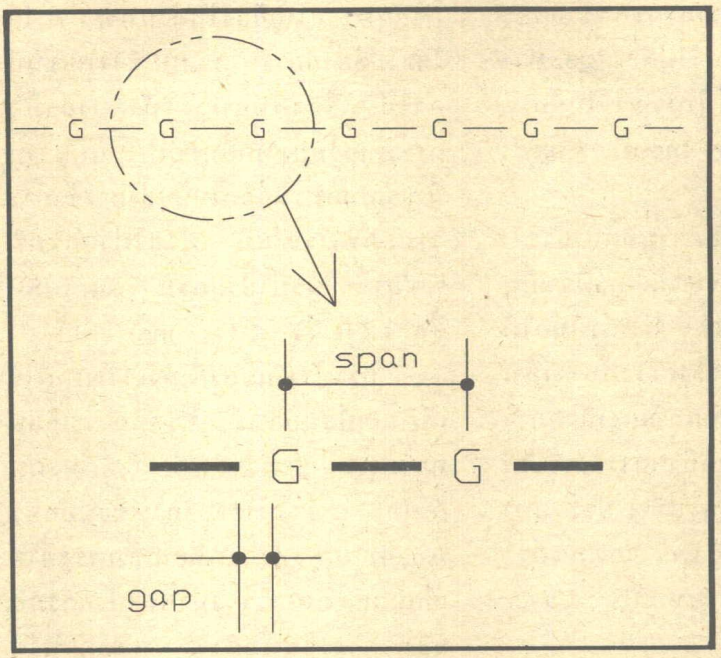
3. อย่าลืมว่าการป้อนขนาดใด ๆ บนแบบที่เขียนในภาวะ Model space ต้องคำนึงถึงสัดส่วนของสเกล

4. คำสั่งนี้จะใช้ Style ปัจจุบันของ Text ที่กำหนดไว้ เป็น Style ของการเขียนอักษร ระวังอย่าตั้ง Style ในแบบความสูงคงที่ เพราะว่าโปรแกรมจะมีกำหนดขนาดอักษรด้วย

5. การยกเลิกคำสั่งขณะที่พรอมต์ใด ๆ ของคำสั่ง ใช้การกด Ctrl+C เช่นเดียวกับการรันคำสั่งปกติใน AutoCAD

6. คำสั่งนี้จะมีการจำค่าที่ป้อนไปแล้ว มาเป็นค่าดีฟอลต์ ในการรันครั้งต่อมา (โดยยังไม่มีการโหลด Drawing ใหม่) ทำให้ค่อนข้างสะดวก เมื่อต้องมีการเขียนข้อความในลักษณะเดียวกันกับหลาย ๆ เส้นขนาน

7. การกด Enter ที่พรอมต์ป้อนค่า โดยยังไม่มีค่าดีฟอลต์ หรือไม่ได้ป้อนค่าใด ๆ เลย จะมีข้อความเตือนว่า Now not a default value. Must be input value, first. และวนกลับมาที่พรอมต์เดิม ให้ป้อนค่าใหม่อีกครั้ง



รูปที่ 3 แสดงการเรียกส่วนต่าง ๆ ของคำสั่ง TINL

Pseudocode ของคำสั่ง TINPL

ส่วนเมนโปรแกรม

1. อ่านค่าตัวแปรระบบที่เกี่ยวข้องและฟังก์ชัน *error* เก็บไว้ที่ตัวแปร oldxxx

2. Redefine ฟังก์ชัน *error* ให้มีการทำงาน reset ค่าตัวแปรระบบกลับค่าเดิม เมื่อมีการยกเลิกการทำงานของคำสั่ง

3. ปิดการแสดงผลของฟังก์ชัน Command

4. **Get user Input, epl1 :** เส้นตรงเส้นแรก

ให้ผู้ใช้เลือกเส้นตรงที่เป็นเส้นหนึ่งของเส้นขนานที่ต้องการ เก็บไว้ที่ตัวแปร epl1 โดยตรวจสอบว่า entity ต้องเป็นชนิด LINE (เส้นตรง)

5. คำนวณหามุมของเส้นตรงที่เลือกด้วย ซับรูทีน angset และฟังก์ชัน angle เก็บไว้ที่ตัวแปร ang1

6. **Get user Input, epl2 :** เส้นตรงเส้นที่สอง

ให้ผู้ใช้เลือกเส้นตรงอีกเส้นที่เหลือที่เป็นเส้นคู่ขนานที่ต้องการ เก็บไว้ที่ตัวแปร epl2 โดยตรวจสอบว่า

6.1 entity ต้องเป็นชนิด LINE

6.2 ไม่ซ้ำกับเส้นตรงเส้นแรกที่เลือกไว้

6.3 เป็นเส้นขนานกับเส้นแรก โดยอาศัยการคำนวณมุมของเส้นตรงด้วยซับรูทีน angset และฟังก์ชัน angle เก็บไว้ที่ตัวแปร ang2 และเปรียบเทียบค่ากับตัวแปร ang1

7. คำนวณหาระยะห่างระหว่างเส้นตรงทั้งสอง เก็บไว้ที่ตัวแปร h โดยใช้ซับรูทีน ponl ปรับจุดของการเลือกจาก epl1 และ epl2 เป็นจุดบนเส้นตรงนั้นจริง ๆ (หรือก็คือการหาโปรเจกชันจุดบนเส้นตรง)

8. **Get user Input, #txth :** ความสูงของตัวอักษร

ให้ผู้ใช้ป้อนความสูงของตัวอักษร โดยตรวจสอบว่าต้องไม่เกินระยะห่างของเส้นตรง (h) ที่คำนวณไว้ในข้อ 7

9. **Get user Input, #txt :** ข้อความที่จะเขียน

ให้ผู้ใช้ป้อนข้อความที่ต้องการ

10. คำนวณพารามิเตอร์ที่จำเป็นในการเขียนตัวอักษร โดยกำหนดให้ข้อความแทรกอยู่ระหว่างกลางเส้นขนานที่เลือก โดยเขียนที่แนวกึ่งกลางของเส้นขนาน ที่จุดกึ่งกลางถัดเข้าไปจากจุดแรกที่เลือก (ดูรูปที่ 1 ประกอบ)

11. อ่านชื่อ Layer ของเส้นตรงแรกที่เลือก และกำหนดเป็น Layer ปัจจุบัน

12. รันฟังก์ชัน command เพื่อใช้คำสั่ง Text เขียนข้อความที่ผู้ใช้ป้อนตามพารามิเตอร์ที่คำนวณไว้ในข้อ 10

13. reset ค่าตัวแปรระบบที่เกี่ยวข้อง และฟังก์ชัน *error* กลับค่าเดิมก่อนรันคำสั่ง TINPL นี้

หมายเหตุ จะเรียกฟังก์ชันที่สร้างขึ้นใหม่ว่า "ซับรูทีน" ส่วนฟังก์ชันที่มีอยู่แล้วใน AutoLISP เอง เรียกว่า "ฟังก์ชัน"

การเขียนตัวอักษรในเส้นตรงแบบอัตโนมัติ

สำหรับท่านที่ทำงานเขียนแบบแปลนที่มีการแสดงแนวท่อของระบบต่าง ๆ เช่น ระบบไฟฟ้า ระบบสุขาภิบาล งานท่อน้ำมัน งานท่อส่งสารเคมีฯ โดยปกติในแนวท่อเรามักจะมีตัวอักษรระบุชื่อระบบ หรือรายละเอียดแสดงไว้เป็นช่วง ๆ บนแนวท่อระบบนั้น ๆ ในแบบแปลน เมื่อเขียนแบบด้วย AutoCAD จะรู้สึกว่าเป็นงานที่ใช้เวลาค่อนข้างมาก ต้องคอยกำหนดระยะการลงทีละช่วง กำหนดมุมการหมุนของตัวอักษร ให้เอียงในด้านที่จะอ่านได้โดยไม่ต้องมีการหมุนแบบ ซึ่งงานเช่นนี้คงไม่ใช่เรื่องสนุกนัก ถ้าต้องลงมือกระทำเองหมดทุกขั้นตอน ในบทความนี้ ขอเสนอทางเลือกใหม่ของการจัดการงานลักษณะเช่นนี้ ด้วยโปรแกรมที่พัฒนาขึ้นในรูปโปรแกรม AutoLISP ชื่อ TINL (Text IN Line)

การทำงานของคำสั่ง TINL ผู้ใช้เพียงแต่ป้อนอักษร หรือคำที่จะเขียน ความสูง ช่องว่างระหว่างอักษรกับเส้นตรง ช่วงห่างของการเขียนอักษร (span) และทำการเลือกเส้นตรงที่ได้เขียนแสดงแนวไว้ เท่านั้นโปรแกรมก็จะทำการเขียนอักษร หรือคำที่ได้กำหนดไว้ พร้อมกับเปิดช่องว่างในเส้นตรงนั้นเองด้วย

ก่อนอื่นขอให้ผู้อ่านได้สร้างไฟล์ TINL.lsp นี้ ด้วยโปรแกรม Text editor ใด ๆ ที่ท่านมีอยู่ ดังแสดงตัวโปรแกรมในรูปที่ 4 จากนั้นทำการโหลดไฟล์ที่บรรทัด Command: ใน AutoCAD ดังนี้

```
Command: (load "c:/lisp/tinl")
```

โดย c:/lisp เป็น Path ที่เก็บไฟล์นี้อยู่ ขอให้สังเกตว่าใช้เครื่องหมาย forward slash (/) เป็นเครื่องหมายบอก Path ต่างจากในดอส ที่ใช้เครื่องหมาย back slash (\) ขอให้ระบุชื่อ Path ที่เก็บให้ตรงด้วย ถ้าไม่มีการพิมพ์ไฟล์ผิดพลาด หรือตกหล่นเลย จะมีข้อความที่บรรทัด Command: เป็น C:TINL ซึ่งจะพร้อมใช้งาน โดยการพิมพ์ TINL โดยตรงที่บรรทัด Command: ได้ สำหรับในกรณีที่พบกับข้อความแสดงความผิดพลาดขอให้กลับไปแก้ไขไฟล์นี้ให้ถูกต้อง แล้วทำการโหลดใหม่อีกครั้ง

ใช้คำสั่ง TINL

เราจะมาทดสอบการทำงานของคำสั่ง TINL ดังนี้

1. เริ่มจากเปิด Drawing ใหม่ แล้วทำการเขียนเส้นตรงเส้นหนึ่ง จากจุด 1, 1 ไปยังจุด 11, 8

รูปที่ 4 ไฟล์โปรแกรม TINL.lsp

```
;;; TINL.lsp
;;; Program to draw a letter or word in a line.
;;; Developed by S.Chatchawal, 1994.
;;;=============================================================
;; Text length
(defun textlm ( e / ptl )
  (setq ptl (textbox (entget e)))(abs(car (mapcar '- (cadr ptl) (car ptl))))
)
;; Read dxf value from an entity.
(defun dxf ( code e) (cdr (assoc code (entget e))))
;; GDDEF (GetDist with DEFault value check subroutine)
;; Argument: a prompt string & a global variabe name.
;; Must be declare a global variable,gvar. Before use GDDEF.
(defun gddef ( prmpt gvar / bye var)
  (while (not bye)
    (princ (strcat "\n  " prmpt ":<"))(princ gvar)(princ "> ")
    (setq var (getdist))
    (if (= var nil)
      (if (/= gvar "")(setq bye T)
        (princ "\n!!! Now not a default value. Must be input value, first.")
      )
      (setq bye T gvar var)
    )
  )
  gvar
)
```

รูปที่ 4 ไฟล์โปรแกรม TINL.lsp (ต่อ)

```
;; GSDEF (GetString with DEFault value check subroutine)
;; Argument: a prompt string & a global variabe name.
;; Must be declare a global variable, gvar. Before use GSDEF.
(defun gsdef ( prmpt gvar / bye var)
  (while (not bye)
    (princ (strcat "\n  " prmpt ":<"))(princ gvar)(princ "> ")
    (setq var (getstring))
    (if (= var "")
      (if (/= gvar "")
        (setq bye T)
        (princ "\n!!! Now not a default value. Must be input value, first.")
      )
      (setq bye T gvar var)
    )
  )
  gvar
)
;; Even no check.
(defun even (no) (if (= 0.0 (rem no 2.0)) no nil) )
;; Angle setup in range [pi/2,0] U (3*pi/2,2*pi).
(defun angset ( e / p1 p2 ang)
  (setq p1 (dxf 10 e) p2 (dxf 11 e) ang (angle p1 p2) )
  (if (or (<= ang (/ pi 2.0))
          (and (> ang (* 3.0 (/ pi 2.0))) (< ang (* 2.0 pi)) )
      )
      (list p1 p2)(list p2 p1)

)
)
;; Mirror point
(defun mirrpt ( pt refpt)(polar refpt (+ (angle refpt pt) pi) (distance refpt pt)))
;; Points to draw text.
(defun ptlmk ( span e / stpt enpt ang cpt ll n0 f d0 ptl cpl lpl pul
                span1/2 pu1 i
             )
  (setq stpt (car (angset e)) enpt (cadr (angset e)) ang (angle stpt enpt))
  (setq cpt (mapcar '/ (mapcar '+ stpt enpt) '(2.0 2.0 2.0)))
  (setq ll (distance stpt enpt); Length of line.
        span (atof (rtos span)); Confirm span to floating data type.
        n0 (fix (/ ll span))   ; no.of span => no.of text=1+ no.of span
        f (- ll (* n0 span))   ; remainder of line distance.
        d0 (* (/ #dead 100.0) span) ; dead band ,#dead % of span.
  )
  (if (and (> f 0.0) (< (/ f 2.0) d0) )(setq n0 (- n0 2)))
  (if (or (< n0 0)(= n0 1)) (setq n0 0))
  (cond
    ((= n0 0) (setq ptl (list cpt))) ; --- X ---
    ((even n0)                       ; --- X -(CL)- X ---
      (setq cpl '())
      (setq span1/2 (/ span 2.0)
            pu1 (polar cpt ang span1/2) pul (list pu1) i 1
      )
      (repeat (1- (/ n0 2))
        (setq pui (polar pu1 ang (* i span))
              pul (append pul (list pui))  i (1+ i)
        )
      )
      (foreach l pul (setq lpl (append (list (mirrpt l cpt)) lpl)))
      (setq ptl (append lpl cpl pul))
    )
    (T                       ;odd ; --- X --- X (CL) --- X ---
      (setq cpl (list cpt))
      (setq pu1 (polar cpt ang span) pul (list pu1) i 1)
      (repeat (1- (/ (1- n0) 2))
        (setq pui (polar pu1 ang (* i span))
              pul (append pul (list pui)) i (1+ i)
        )
      )
      (foreach l pul (setq lpl (append (list (mirrpt l cpt)) lpl)))
      (setq ptl (append lpl cpl pul))
    )
  )
  ptl
)
;; Main progrom
(setq #txt "" #txth "" #dead "" #span "" #gap "")
(defun c:tinl (/ oldcmd oldlay oldtxt oldblp olderr stop e p1 p2
               ll stpt enpt ang etxt txtl ptl pu pl lptl i
              )
  (setq oldcmd (getvar "cmdecho") oldlay (getvar "clayer")
        oldtxt (getvar "texteval") oldblp (getvar "blipmode")
        olderr *error*
  )
  (defun *error* (msg)
    (if (not (member msg '("console break" "Function cancelled")))
      (princ (strcat "Error: " msg))
    )
    (command "undo" "e" "u")
    (command "layer" "s" oldlay "")(setvar "texteval" oldtxt)
    (setvar "blipmode" oldblp)(setvar "cmdecho" oldcmd)
    (setq *error* olderr)(princ)
  )
  (setvar "cmdecho" 0)
  (command "undo" "g")
  (while (not stop)
    (while (not (setq e (car (entsel)))))
    (if (= "LINE" (dxf 0 e))(setq stop T)
      (princ "\n The entity must be a line. \n Try again.")
    )
  )
  (setq #txt (gsdef "Text" #txt) #txth (gddef "Text Height" #txth))
  (setq stop nil)
  (while (not stop)
    (initget 6)
    (setq #dead (gddef "Percent of Dead band (1-100)" #dead))
    (if (and (> #dead 0.0) (<= #dead 100.0))
      (setq stop T)
      (princ "\n This percent value must not exceeds 100. \n Try again.")
    )
  )
  (setq p1 (dxf 10 e) p2 (dxf 11 e) ll (distance p1 p2) stop nil)
  (while (not stop)
    (setq #span (gddef "Span" #span))
    (if (< #span ll)(setq stop T)
      (princ "\n Span exceeds line length!!! \n Try again.")
    )
  )
  (setq stop nil)
  (while (not stop)
    (setq #gap (gddef "Gap" #gap))
    (if (< #gap (* 0.2 #span)) (setq stop T)
      (princ "\n Gap exceeds 20% of Span!!! \n Try again.")
    )
  )
  (setq stpt (car (angset e)) enpt (cadr (angset e)) ang (angle stpt enpt))
  (setvar "blipmode" 0)(setvar "texteval" 1)
  (command "text" "0,0" #txth 0.0 #txt)
  (setq etxt (entlast))(setq txtl (textlm etxt))(entdel etxt)
;; Point to draw text.
  (setq ptl (ptlmk #span e))


;; Draw text.
  (entdel e) ; Erase original line.
  (command "layer" "s" (dxf 8 e) "")
  (foreach l ptl (command "text" "mc" l #txth (angtos ang 0 2) #txt))
  (setvar "texteval" oldtxt)
;; Draw line.
  (setq adj (/ (+ (* 2.0 #gap) txtl) 2.0))
  (foreach l ptl
    (setq pu (polar l ang adj) pl (mirrpt pu l)
          lptl (append lptl (list pl pu))
    )
  )
  (setq lptl (append lptl (list enpt)) i 0)
  (repeat (1+ (length ptl))
    (command "line" stpt (nth i lptl) "")
    (setq i (1+ i) stpt (nth i lptl) i (1+ i))
  )
  (command "layer" "s" oldlay "")
  (setvar "blipmode" oldblp)
  (command "undo" "e")
  (setvar "cmdecho" oldcmd)
  (setq *error* olderr)(princ)
)
;-------------------------------------------------------------------------
```

2. ในตอนนี้ถ้ายังไม่มีโหลดไฟล์ TINL ให้กระทำก่อนตามวิธีการที่ได้กล่าวไว้ในข้างต้น จากนั้นที่บรรทัด Command: พิมพ์ TINL กด Enter* (*ต่อไปจะละคำว่า "กด Enter" ออก ซึ่งผู้อ่านจะต้องกระทำด้วยทุกครั้ง เมื่อป้อนค่าครบแล้ว)

3. จะมีพรอมต์

```
Select object:
```

ให้เลือกเส้นตรงที่ได้วาดขึ้น

ถ้าการเลือกไม่ถูกวัตถุใด ๆ เลย โปรแกรมจะยังคงวนมาที่พรอมต์ Select object: จนกว่าจะเลือกถูกวัตถุ และถ้าวัตถุที่เลือกไม่ใช่รูปวาดชนิดเส้นตรง (Line) จะมีข้อความเตือนว่า The entity must be a LINE. Try again. และวนกลับมาที่พรอมต์เดิม ให้เลือกใหม่อีกครั้ง เป็นเช่นนี้จนกว่าจะเลือกวัตถุ ได้ถูกต้อง

4. จะมีพรอมต์

```
Text: < >
```

ให้ป้อนตัวอักษร หรือคำ ที่ต้องการเขียนในเส้นตรงนี้ ในที่นี้ป้อน A

5. จะมีพรอมต์

```
Text Height: < >
```

ให้ป้อนความสูงของอักษร ในที่นี้ป้อน 0.2

6. จะมีพรอมต์

```
Percent of Dead band (1-100): < >
```

ให้ป้อนค่าเปอร์เซ็นต์ของระยะที่จะยอมให้มีการเขียนตัวอักษร หรือคำที่บริเวณปลายเส้น โดยเป็นเปอร์เซ็นต์เทียบกับระยะห่างของช่วงการเขียนอักษรนี้ ในที่นี้ป้อน 10

ถ้าป้อนค่าเกิน ช่วง 1-100 จะมีข้อความเตือนว่า This percent value must not exceeds 100. Try again. และวนกลับไปที่ พรอมต์เดิมอีกครั้ง ให้ป้อนค่าใหม่ เป็นเช่นนี้จนกว่าจะถูกต้อง

7. จะมีพรอมต์

```
Span: < >
```

ให้ป้อนระยะช่วงห่างของการเขียนอักษร ในที่นี้ป้อน 1

ถ้าค่าที่ป้อนเกินความยาวของเส้นตรงที่เลือก จะมีข้อความเตือนว่า Span exceeds line length!!! Try again. และวนกลับมาที่พรอมต์เดิม ให้ป้อนค่าใหม่อีกครั้ง เป็นเช่นนี้จนกว่าจะถูกต้อง

8. จะมีพรอมต์

```
Gap: < >
```

ให้ป้อนระยะของช่องว่างระหว่างเส้นตรงกับอักษร ในที่นี้ป้อน 0.1

ซึ่งค่าที่ป้อนนี้ยอมให้มีระยะได้ไม่เกิน 20% ของค่า Span ถ้าค่าเกินจากนี้ จะมีข้อความเตือนว่า Gap exceeds 20% of Span!!! Try again. และวนกลับมาที่พรอมต์เดิม ให้ป้อนค่าใหม่อีกครั้ง เป็นเช่นนี้จนกว่าจะถูกต้อง

จากนั้นโปรแกรมจะทำการเขียนตัวอักษร หรือคำที่ป้อน ตามลักษณะที่เรากำหนดไว้ พร้อมกับเปิดช่องว่างระหว่างตัวอักษร กับเส้นตรงให้ด้วย

ข้อสังเกต

1. ตัวอักษร หรือคำที่เขียนจะอยู่บน Layer เดียวกับเส้นตรงที่เลือก

2. ไม่ว่าเส้นตรงจะอยู่ในแนวใดๆ โปรแกรมจะกำหนดการเขียนอักษรเฉพาะมุมในช่วง [0, pi/2] กับ (−pi/2,0) (เป็นมุมในหน่วยเรเดียน) ทั้งนี้เพื่อให้อ่านแบบได้โดยไม่ต้องมีการหมุนแบบ

3. อย่าลืมว่าการป้อนขนาดใด ๆ บนแบบที่เขียนในภาวะ Model space ต้องคำนึงถึงสัดส่วนของสเกล

4. คำสั่งนี้จะใช้ Style ปัจจุบันของ Text ที่กำหนดไว้ เป็น Style ของการเขียนอักษร ระวังอย่าตั้ง Style ในแบบความสูงคงที่ เพราะว่าโปรแกรมจะมีการกำหนดขนาดอักษรด้วย

5. การยกเลิกคำสั่งขณะที่พรอมต์ใด ๆ ของคำสั่ง ใช้การกด Ctrl+C เช่นเดียวกับการรันคำสั่งปกติใน AutoCAD

6. ภายหลังการรันคำสั่งจบไปแล้ว การยกเลิกด้วยคำสั่ง U จะเป็นการยกเลิกการทำงานทั้งหมดของคำสั่งกล่าวคือจะได้เส้นตรงเดิมที่ยังไม่มีการเขียนส่วนใด ๆ เพิ่มเลย

7. คำสั่งนี้จะมีการจำค่าที่ป้อนไปแล้ว มาเป็นค่าดีฟอลต์ ในการรันครั้งต่อมา (โดยยังไม่มีการโหลด Drawing ใหม่) ทำให้ค่อนข้างสะดวกเมื่อต้องมีการเขียนข้อความในลักษณะเดียวกัน กับหลาย ๆ เส้นตรง

8. การกด Enter ที่พรอมต์ป้อนค่า โดยยังไม่มีค่าดีฟอลต์ หรือไม่ได้ป้อนค่าใด ๆ เลย จะมีข้อความเตือนว่า Now not a default value. Must be input value, first. และวนกลับมาที่พรอมต์เดิม ให้ป้อนค่าใหม่อีกครั้ง

Pseudocode ของคำสั่ง TINL

ส่วนเมนโปรแกรม

1. อ่านค่าตัวแปรระบบที่เกี่ยวข้องและฟังก์ชัน *error* เก็บไว้ที่ตัวแปร oldxxx

2. Redefine ฟังก์ชัน *error* ให้มีการทำงาน reset ค่าตัวแปรระบบกลับค่าเดิม เมื่อมีการยกเลิกการทำงานของคำสั่ง

3. ปิดการแสดงผลของฟังก์ชัน Command

4. **Get user input, e :** เส้นตรง

ให้ผู้ใช้เลือก entity เส้นตรง โดยจะมีการตรวจสอบว่าต้องเป็น entity ชนิดเส้นตรง

5. **Get user input, #txt :** ข้อความที่จะเขียน

ให้ผู้ใช้ป้อนข้อความที่จะเขียน รับเฉพาะคำเท่านั้น หรือไม่ยอมให้มีการเว้นวรรค

6. **Get user input, #txth :** ความสูงของตัวอักษร

ให้ผู้ใช้ป้อนความสูงของตัวอักษร

7. **Get user input, #dead :** ระยะบริเวณปลายของเส้นตรงที่ยอมให้เขียนข้อความ

ให้ผู้ใช้ป้อนค่าเป็นตัวเลขเปอร์เซ็นต์ของความยาวเส้นตรง โดยจะมีการตรวจสอบว่าเป็นตัวเลขในช่วง 0-100

8. คำนวณหาความยาวของเส้นตรง เก็บไว้ที่ตัวแปร ll

9. **Get user input, #span :** ความยาวของช่วงห่างการเขียนข้อความ

ให้ผู้ใช้ป้อนค่าระยะห่างของการเขียนข้อความ โดยจะมีการตรวจสอบว่าต้องไม่เกินความยาวของเส้นตรง (ll)

10. **Get user input, #gap :** ช่องว่างระหว่างเส้นตรงที่เปิดออกกับข้อความที่เขียน

ให้ผู้ใช้ป้อนค่าเป็นตัวเลขระยะห่างของเส้นตรงกับข้อความ โดยจะมีการตรวจสอบว่าค่าต้องไม่เกิน 20% ของค่า #span

11. คำนวณหาความยาว (เชิงเส้น) ของข้อความที่จะเขียน โดยสร้าง entity ชั่วคราวที่เป็น text ของข้อความนี้ แล้วใช้ซับรูทีน textlm คำนวณค่าแล้วลบ entity ชั่วคราวนี้ออก

12. คำนวณจุดการเขียนทั้งหมดด้วยซับรูทีน ptlmk เก็บไว้ที่ตัวแปร ptl

13. ลบเส้นตรงเดิม (e) ออก

14. อ่านชื่อ Layer ของเส้นตรงที่เลือกไว้ และกำหนดเป็น Layer ปัจจุบัน

15. เขียนข้อความ (#txt) ตามตำแหน่งของจุดในตัวแปร ptl

16. เขียนเส้นตรงย่อย ๆ ตามแนวเส้นตรงเดิม โดยคำนวณจุดการเขียนจากจุดใน ptl

17. reset ค่าตัวแปรระบบที่เกี่ยวข้อง กับฟังก์ชัน *error* กลับค่าเดิมก่อนรันคำสั่ง TINL นี้

การเปลี่ยน CASE ของข้อความเป็นพิมพ์ใหญ่หรือเล็ก

การเขียนแบบใน AutoCAD ส่วนที่เกี่ยวกับการเขียนอักษร หรือข้อความ เมื่อต้องการแก้ไขใด ๆ จะกระทำได้โดยการใช้คำสั่ง DDEDIT, DDMODIFY หรือ CHANGE ซึ่งเป็นคำสั่งพื้นฐานใน AutoCAD เอง โดยการทำงานหลังจากเราเลือก entity หรือรูปวาดที่เป็นข้อความที่จะแก้ไขแล้ว จะมีพรอมต์ให้ตอบค่าใหม่ที่ต้องการ ซึ่งในกรณีที่ต้องการเปลี่ยนแปลงเฉพาะ CASE ของตัวอักษร ให้เป็นตัวพิมพ์ใหญ่ หรือเล็กทั้งหมดอย่างใดอย่างหนึ่ง ก็ต้องเสียเวลาแก้ไขพิมพ์ค่าใหม่ทั้งหมด ในบทความนี้ขอเสนอคำสั่ง TCASE ที่พัฒนาขึ้นจากการโปรแกรม AutoLISP ซึ่งจะมีการทำงานที่ง่ายกว่าการใช้คำสั่งพื้นฐานใน AutoCAD เอง กล่าวคือไม่ต้องมีการพิมพ์ใหม่ทั้งหมด เพียงแต่ตอบพรอมต์ของคำสั่งในแบบที่ต้องการเท่านั้น โปรแกรมก็จะทำการเปลี่ยนแปลงอักษรให้มีรูปแบบใน CASE ที่ต้องการ

ก่อนอื่นขอให้ผู้อ่านได้สร้างไฟล์ TCASE.lsp นี้ ด้วยโปรแกรม Text editor ใด ๆ ที่ท่านมีอยู่ตามรูปที่ 6 จากนั้นทำการโหลดไฟล์ที่บรรทัด Command: ใน AutoCAD ดังนี้

```
Command : (load "C:/lisp/tcase")
```

โดย c:/lisp เป็น Path ที่เก็บไฟล์นี้อยู่ ขอให้สังเกตว่าใช้เครื่องหมาย forward slash (/) เป็นเครื่องหมายบอก Path ต่างจากในดอส ที่ใช้เครื่องหมาย back slash (\) ขอให้ระบุชื่อ Path ที่เก็บให้ตรงด้วย ถ้าไม่มีการพิมพ์ไฟล์ผิดพลาด หรือตกหล่นเลย จะมีข้อความที่บรรทัด Command: เป็น C:TCASE ซึ่งจะพร้อมใช้งานโดยการพิมพ์ TCASE โดยตรงที่บรรทัด Command: ได้ สำหรับในกรณีที่พบกับข้อความแสดงความผิดพลาดขอให้กลับไปแก้ไขไฟล์นี้ให้ถูกต้องแล้วทำการโหลดใหม่อีกครั้ง

ใช้คำสั่ง TCASE

เราจะมาทดสอบการทำงานของคำสั่ง TCASE ดังนี้

1. เริ่มจากเปิด Drawing ใหม่ แล้วใช้คำสั่ง TEXT ทำการเขียนข้อความ Test for Tcase. ตามบรรทัดแรกในรูปที่ 5 (ให้มี Case ของอักษรตรงตามนี้ด้วย)

2. ในตอนนี้ถ้ายังไม่มีโหลดไฟล์ TCASE ให้กระทำก่อนตามวิธีการที่ได้กล่าวไว้ในข้างต้น จากนั้นที่บรรทัด Command: พิมพ์ TCASE กด Enter*

(*ต่อไปจะละคำว่า "กด Enter" ออก ซึ่งผู้อ่านจะต้องกระทำด้วยทุกครั้ง เมื่อป้อนค่าครบแล้ว)

3. จะมีพรอมต์

```
Lower/<Upper>
```

เป็นตัวเลือกว่าจะข้อความที่จะแปลงต่อไปนี้ จะให้เป็นแบบตัวพิมพ์เล็ก (lower) หรือตัวพิมพ์ใหญ่ (Upper) ทั้งหมดอย่างใดอย่างหนึ่ง สำหรับในที่นี้ให้ทดลองเลือก . Upper โดยการกด Enter เนื่องจากเป็นค่าดีฟอลต์แล้ว

4. จะมีพรอมต์

```
All/layer/<Select>:
```

ถ้าเลือก All โปรแกรมจะทำการเปลี่ยนแปลงทุก entities ที่เป็นตัวอักษรให้เป็น Case ตามที่เรากำหนดไว้ในข้อ 3. ถ้าเลือก Layer จะมีพรอมต์ Layer: ให้ป้อนชื่อ Layer ของตัวอักษรที่ต้องการเปลี่ยน Case นี้ ในที่นี้ให้เลือกตัวเลือก Select ซึ่งกระทำโดยการกด Enter เพื่อรับค่าดีฟอลต์นี้

5. จะมีพรอมต์

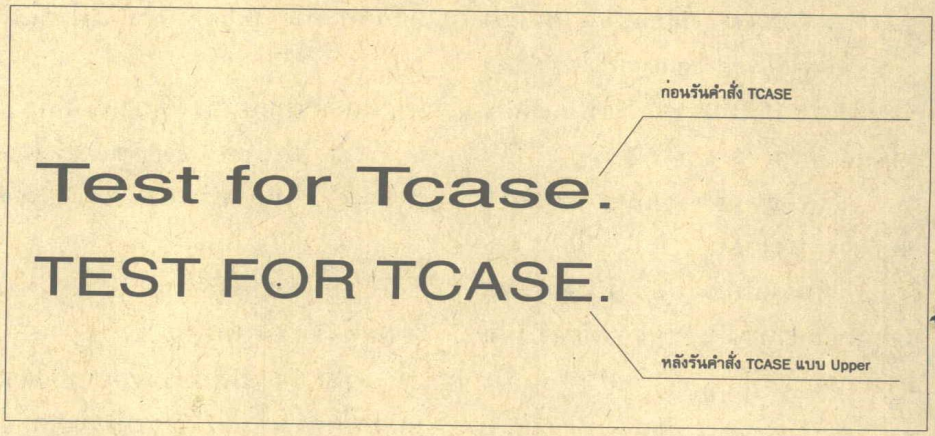
```
Select objects:
```

ที่พรอมต์นี้ให้เลือกอักษรที่ต้องการจะเปลี่ยน Case ตามที่เรากำหนดไว้ในข้อ 3. ในที่นี้ให้เลือกข้อความที่ได้เขียนไว้

```
Select objects:
```

ให้กด Enter เมื่อเลือก entities ที่ต้องการครบแล้ว ในที่นี้มี entity เดียวให้กด Enter

6. โปรแกรมจะมีการเปลี่ยน Case เป็นแบบที่เรากำหนดไว้ ในที่ข้อความที่เราป้อนไว้จะเป็นตัวพิมพ์ใหญ่ทั้งหมด เป็น TEST FOR TCASE ตามรูปที่ 5 ในบรรทัดที่สอง และมีข้อความแสดงว่าได้ทำงานเสร็จแล้ว เป็น Change text case to Upper, completely.



รูปที่ 5 การใช้คำสั่ง TCASE

การแก้ไขข้อความ

การเขียนแบบใน AutoCAD เมื่อเกี่ยวข้องกับการเขียนข้อความ เป็นไปได้ที่บางครั้งเราอาจจะมีการสะกดคำศัพท์ผิดไป ซึ่งอาจจะผิดเหมือน ๆ กันหมด หรือเป็นส่วนใหญ่ของข้อความในแบบ ถ้าต้องมาแก้ไขทีละประโยคจนครบทั้งหมด ดูเหมือนว่าจะไม่ใช่วิสัยของการใช้โปรแกรม AutoCAD เลย เนื่องจาก AutoCAD ได้เตรียมเครื่องมือให้เราสามารถสร้างคำสั่งขึ้นใหม่เพื่อช่วยงานเฉพาะหนึ่ง ๆ เพิ่มจากคำสั่งพื้นฐานที่มีอยู่แล้ว สำหรับในบทความนี้ ขอเสนอคำสั่ง TREPLACE ซึ่งพัฒนาโดยอาศัยเครื่องมือที่เรียกว่า การโปรแกรม AutoLISP

การทำงานของคำสั่ง TREPLACE ผู้ใช้เพียงแต่ป้อนข้อความ หรือคำที่ต้องการแก้ไข แล้วตามด้วยข้อความหรือคำใหม่ และเลือกชนิดของการแก้ไขตามต้องการ จากนั้นโปรแกรม จะทำการแทนที่ (Replace) ข้อความหรือคำเก่าด้วยข้อความ หรือคำใหม่ที่เราได้ป้อนไว้ จนครบตามที่กำหนดเอง ไม่ต้องมาเสียเวลาแก้ไขเองทีละตัวเลย

ก่อนอื่นขอให้ผู้อ่านได้สร้างไฟล์ TREPLACE.lsp นี้ ด้วยโปรแกรม Text editor ใด ๆ ที่ท่านมีอยู่ตามรูปที่ 7 จากนั้นทำการโหลดไฟล์ที่บรรทัด Command: ใน AutoCAD ดังนี้

```
Command : (load "C:/lisp/treplace")
```

โดย c:/lisp เป็น Path ที่เก็บไฟล์นี้อยู่ ขอให้สังเกตว่าใช้เครื่องหมาย forward slash (/) เป็นเครื่องหมายบอก Path ต่างจากในดอส ที่ใช้เครื่องหมาย back slash (\) ขอให้ระบุชื่อ Path ที่เก็บให้ตรงด้วย ถ้าไม่มีการพิมพ์ไฟล์ผิดพลาดหรือตกหล่นเลย จะมีข้อความที่บรรทัด Command: เป็น C:TREPLACE ซึ่งจะพร้อมใช้งาน โดยการพิมพ์ TREPLACE โดยตรงที่บรรทัด Command: ได้ สำหรับในกรณีที่พบกับข้อความแสดงความผิดพลาด ขอให้กลับไปแก้ไขไฟล์นี้ให้ถูกต้อง แล้วทำการโหลดใหม่อีกครั้ง

ใช้คำสั่ง TREPLACE

เราจะมาทดสอบการทำงานของคำสั่ง TREPLACE ดังนี้

รูปที่ 6 ไฟล์โปรแกรม TCASE.lsp

```
;;; TCASE.lsp, Change Text CASE to either Lower or Upper case.
;;; Developed by S.Chatchawal, 1994.
;;;=============================================================
;; G_sys ; get system variables.
;; Should be set "cmdecho" to last element in sysl argument.
(defun g_sys ( sysl / sl)
  (foreach l sysl (setq sl (append sl (list (cons l (getvar l))))))
  (append (list *error*) sl)
)
;; R_sys ; reset system variables.
(defun r_sys ( oldsys )
  (setq *error* (car oldsys))
  (foreach l (cdr oldsys)
    (if (= l "clayer") (command "layer" "s" (cdr l) "")
      (setvar (car l) (cdr l))
    )
  )(princ)
)
;; Errmsg
(defun errmsg (msg)
  (if (not (member msg '("console break" "Function cancelled")))
    (princ (strcat "Error: " msg))
  )
)
;; Intersection of selection set s1 and s2.
(defun sinter (s1 s2 / smax1 smax2 s3 big small i ei)
  (setq smax1 (sslength s1) smax2 (sslength s2) s3 (ssadd) i -1)
  (if (>= smax1 smax2) (setq big s1 small s2)(setq big s2 small s1))
  (repeat (sslength small)
    (setq ei (ssname small (setq i (1+ i))))(if (ssmemb ei big) (ssadd ei s3))
  )
  s3
)
;;-------------------------------------------------------------------
;; Main program
(defun c:tcase (/ oldsys s ss case ans stop laynm i ei txt c)
  (setq oldsys (g_sys '("cmdecho")))
  (defun *error* (msg)
    (errmsg msg)(command "undo" "e" "u")(r_sys oldsys)(princ)
  )
  (setvar "cmdecho" 0)(command "undo" "g")
  (initget "Lower Upper")
  (setq case (getkword "\n Lower/<Upper>: "))
  (if (not case) (setq case "Upper"))
  (initget "All Layer Select")
  (setq ans (getkword "\n All/Layer/<Select>: "))
  (if (not ans)(setq ans "Select"))
  (cond
    ((= ans "All")
      (setq s (ssget "x" (list (cons 0 "TEXT"))))
    )

   ((= ans "Layer")
      (setq stop nil)
      (while (not stop)
       (setq laynm (getstring "\n Layer: "))
       (if (tblsearch "layer" laynm)(setq stop T)
         (princ (strcat "\n " laynm " not found. \n Try again."))
       )
      )
      (setq s (ssget "x" (list (cons 0 "TEXT")(cons 8 laynm))))
    )
    ((= ans "Select")
      (setq ss (ssget) s (sinter ss (ssget "x" (list (cons 0 "TEXT")))))
    )
  )
  (cond ((= case "Upper")(setq c nil))((= case "Lower")(setq c T)))
  (setq i -1)
  (repeat (sslength s)
    (setq ei (ssname s (setq i (1+ i))) txt (cdr (assoc 1 (entget ei))))
    (entmod (subst (cons 1 (strcase txt c))(cons 1 txt)(entget ei)))
  )
  (princ (strcat "\n Change text case to " case ", completely."))
  (r_sys oldsys)(command "undo" "e")(princ)
)
```

1. เริ่มจากเปิด Drawing ใหม่ แล้วรันคำสั่ง TEXT เขียนข้อความดังนี้

This cat is on the table & the dog is on the floor.

2. ในตอนนี้ถ้ายังไม่มีโหลดไฟล์ TREPLACE ให้กระทำก่อนตามวิธีการที่ได้กล่าวไว้ในข้างต้น จากนั้นที่บรรทัด Command: พิมพ์ TREPLACE กด Enter*

(*ต่อไปจะละคำว่า "กด Enter" ออก ซึ่งผู้อ่านจะต้องกระทำด้วยทุกครั้ง เมื่อป้อนค่าครบแล้ว)

3. จะมีพรอมต์

Find:

ให้ป้อนข้อความที่ต้องการจะเปลี่ยนแปลง ในที่นี้ป้อน &

4. จะมีพรอมต์

Replace to:

ให้ป้อนข้อความใหม่ที่ต้องการ ในที่นี้ป้อน and

5. จะมีพรอมต์

letter case, Match/<Whole>:

ตัวเลือก Match หมายถึงว่าจะค้นหาข้อความที่จะเปลี่ยนแปลงตามที่กำหนดไว้ในพรอมต์ Find: โดยสนใจการเป็นตัวอักษรพิมพ์เล็ก หรือใหญ่ด้วย เช่น ถ้าที่ Find: ป้อน cat และเลือก Match ถ้ามีคำว่า Cat ในข้อความจะถือว่าไม่ใช่ cat สำหรับตัวเลือก Whole จะค้นหาโดยไม่สนใจ Case ของตัวอักษร ขอให้สะกดตรงกันเท่านั้น ในที่นี้กด Enter เพื่อรับค่าดีฟอลต์นี้

6. จะมีพรอมต์

All/Layer/Select/<Find & verify>:

ตัวเลือก All หมายถึงให้ค้นหาทุก entities ในแบบปัจจุบันที่เป็นตัวอักษร ตัวเลือก Layer จะมีพรอมต์ Layer: ให้ป้อนชื่อ Layer ที่จะใช้ในการค้นหา ซึ่งจะกระทำเฉพาะบน Layer ที่ป้อน ตัวเลือก Select จะมีพรอมต์ Select objects: ให้เลือก entities ตามต้องการ

และสำหรับตัวเลือก Find&verify โปรแกรมจะทำการค้นหาให้เอง และเมื่อพบจะมีการแสดง Highlight (entity ที่พบจะถูกแสดงเป็นเส้นประ) และมีพรอมต์ Verify Skip/<Change>: เมื่อเลือก Skip จะข้ามข้อความที่พบผ่านไป แต่ถ้าเลือก Change ซึ่งเป็นค่าดีฟอลต์ โปรแกรมจะทำการเปลี่ยนตามค่าที่ป้อนไว้

ในที่นี้เลือก All

7. โปรแกรมจะทำการเปลี่ยน & เป็นคำว่า and และมีพรอมต์ว่า

Replace text, completely.

8. ต่อไปเราจะลองทดสอบตัวเลือก Find&verify ดังนี้

Command: พิมพ์ TREPLACE

Find: พิมพ์ is

Replace to: พิมพ์ was

letter case, Match/<Whole>: กด Enter

All/Layer/Select/<Find & verity>: กด Enter

Verity Skip/<Change>: จะเห็นว่ามี Highlight บนข้อความ ซึ่งหมายความว่าโปรแกรมพบข้อความที่ต้องการให้หา ตามที่ระบุไว้ที่ Find: ถ้าต้องการเปลี่ยนให้กด Enter เพื่อรับค่าดีฟอลต์ Change แต่ถ้าไม่ต้องการเปลี่ยนให้ป้อน S เพื่อ Skip ข้ามที่พบไปโดยไม่มีการเปลี่ยนแปลง

ในที่นี้ให้กด Enter จะเห็นว่าโปรแกรมมีการเปลี่ยนค่าให้จาก is เป็น was และมีพรอมต์

Verify Skip/<Change>:

เนื่องจากโปรแกรมพบคำว่า is อีกในข้อความ (ในขณะนี้คือ is หลังคำว่า dog) ให้ลองเลือก S เพื่อข้ามคำที่พบนี้ไป โดยไม่ต้องมีการเปลี่ยนแปลงใด ๆ จากนั้นจะมีพรอมต์

Replace text, completely.

เป็นอันจบขั้นตอนการใช้งานคำสั่ง TREPLACE ขอให้ผู้อ่านได้ทดลองการใช้ตัวเลือกอื่น ๆ ด้วย

สรุป

บทความนี้ นอกจากจะแสดงถึงตัวโปรแกรมของคำสั่งช่วยในงานเขียนข้อความใน AutoCAD ไว้ให้ใช้งานได้โดยตรงแล้ว คงจะทำให้ผู้อ่านเล็งเห็นถึงประโยชน์ของการดัดแปลง หรือปรับปรุง (Customization) AutoCAD โดยการโปรแกรม AutoLISP ให้มีคำสั่งที่ตรงเฉพาะกับงานลักษณะหนึ่ง ๆ

รูปที่ 7 ไฟล์โปรแกรม TREPLACE.lsp

```
;;; TREPLACE.lsp, Replace phrase or word in Statement.
;;; Developed by S.Chatchawal, 1994.
;;=============================================================================
;; G_sys ; get system variables.
;; Should be set "cmdecho" to last element in sysl argument.
(defun g_sys ( sysl / sl)
  (foreach l sysl (setq sl (append sl (list (cons l (getvar l))))))
  (append (list *error*) sl)
)
;; R_sys ; reset system variables.
(defun r_sys ( oldsys )
  (setq *error* (car oldsys))
  (foreach l (cdr oldsys)
    (if (= l "clayer") (command "layer" "s" (cdr l) "")
      (setvar (car l) (cdr l))
    )
  )(princ)
)
;; Errmsg
(defun errmsg (msg)
  (if (not (member msg '("console break" "Function cancelled")))
    (princ (strcat "Error: " msg))
  )
)
;; Intersection of selection set s1 and s2.
(defun sinter (s1 s2 / smax1 smax2 s3 big small i ei)
  (setq smax1 (sslength s1) smax2 (sslength s2) s3 (ssadd) i -1)
  (if (>= smax1 smax2) (setq big s1 small s2)(setq big s2 small s1))
  (repeat (sslength small)
    (setq ei (ssname small (setq i (1+ i))))(if (ssmemb ei big) (ssadd ei s3))
  )
  s3
)
;; Read dxf
(defun dxf (code e) (cdr (assoc code (entget e))))
;; Replace
;;ex1: (repl "dog" "cat" e 0)
;; If the old text in e is "This is a Dog." ,the new text is "This is a cat.".
;;ex2: (repl "dog" "cat" e 1)
;; Same entity as above, This example returns "This is a cat.".
;; Because the case argument is specified to match case (1).
(defun repl (old new e case / txt oldlen result i chk process)
  (setq txt (dxf 1 e) oldlen (strlen old) result "" i 1)

(repeat (strlen txt)
    (setq chk (substr txt i oldlen))
    (cond ; Method to check separate by a case argument.
      ((= case 0) ; Whole case
        (if (= (strcase chk) (strcase old))
          (setq process 1)(setq process 0)
        )
      )
      ((= case 1) ; Match case
        (if (= chk old) (setq process 1)(setq process 0))
      )
    )
```

→

รูปที่ 7 ไฟล์โปรแกรม TREPLACE.lsp (ต่อ)

```
    (cond
      ((= process 1)  ; Update text.
        (setq result (strcat result new) i (+ i oldlen))
      )
      ((= process 0)  ; Skip 1 letter.
        (setq result (strcat result (substr chk 1 1)) i (1+ i))
      )
    )
  )
  (entmod (subst (cons 1 result)(cons 1 txt) (entget e)))
)
;; Find & Verify.
(defun fi&ver ( old new e case
          / txt oldlen newlen result i chk process ans upd
         )
  (setvar "highlight" 1)
  (setq txt (dxf 1 e) oldlen (strlen old) newlen (strlen new) result "" i 1)
  (repeat (strlen txt)
    (setq txt (dxf 1 e) chk (substr txt i oldlen))
    (cond ; Method to check separate by a case argument.
      ((= case 0) ; Whole case
        (if (= (strcase chk) (strcase old))
          (setq process 1)(setq process 0)
        )
      )
      ((= case 1) ; Match case
        (if (= chk old) (setq process 1)(setq process 0))
      )
    )
    (cond
      ((= process 1)  ; Update text.
        (redraw e 3)
        (initget "Skip Change")
        (setq ans (getkword "\n Verify Skip/<Change>: "))
        (if (not ans)(setq ans "Change"))
        (cond
          ((= ans "Change")
            (setq result (strcat result new)
                tail (substr txt (+ i oldlen)
                         (1+ (- (strlen txt) (+ i oldlen)))
                         )
                i (+ i newlen)
                upd (strcat result tail)
            )
            (entmod (subst (cons 1 upd)(cons 1 txt)(entget e)))
          )
          ((= ans "Skip")
            (setq result (strcat result (substr chk 1 1)) i (1+ i))
          )
        )
        (redraw e 4)
      )
      ((= process 0)  ; Skip 1 letter.
        (setq result (strcat result (substr chk 1 1)) i (1+ i))
      )
    )
  )
)
;;-------------------------------------------------------------------
;; Main program
(defun c:treplace (/ oldsys s ss case ans stop laynm i ei txt c)
  (setq oldsys (g_sys '("cmdecho" "highlight")))
  (defun *error* (msg)
    (errmsg msg)(command "undo" "e")(r_sys oldsys)(princ)
  )
  (setvar "cmdecho" 0)(command "undo" "g")
  (setq old (getstring T "\n Find: ")
      new (getstring T "\n Replace to: ")
  )
  (initget "Whole Match")
  (setq case (getkword "\n letter case, Match/<Whole>: "))
  (if (not case) (setq case "Whole"))
  (cond ((= case "Whole")(setq case 0))
      ((= case "Match")(setq case 1))
  )
  (initget "All Layer Select Find&verify")
  (setq ans (getkword "\n All/Layer/Select/<Find&verify>: "))
  (if (not ans)(setq ans "Find&verify"))
  (cond
    ((= ans "All")
        (setq s (ssget "x" (list (cons 0 "TEXT"))) i -1)
        (repeat (sslength s)(repl old new (ssname s (setq i (1+ i))) case))
    )
    ((= ans "Layer")
        (setq stop nil)
        (while (not stop)
          (setq laynm (getstring "\n Layer: "))
          (if (tblsearch "layer" laynm)(setq stop T)
            (princ (strcat "\n " laynm " not found. \n Try again."))
          )
        )
(setq s (ssget "x" (list (cons 0 "TEXT")(cons 8 laynm))) i -1)
        (repeat (sslength s)(repl old new (ssname s (setq i (1+ i))) case))
    )
    ((= ans "Select")
        (setq ss (ssget) s (inter ss (ssget "x" (list (cons 0 "TEXT")))) i -1)
        (repeat (sslength s)(repl old new (ssname s (setq i (1+ i))) case))
    )
    ((= ans "Find&verify")
        (setq s (ssget "x" (list (cons 0 "TEXT"))) i -1)
        (repeat (sslength s)
          (setq e (ssname s (setq i (1+ i))))
          (fi&ver old new e case)
        )
    )
  )
  (princ "\n Replace text, completely.")
  (command "undo" "e")(r_sys oldsys)(princ)
)
```